NAME: Daniel Gonzalez Cedre NETID: —

Exam 2.p.fa.2024

Principles of Computing

11th of December, 2024

UNIVERSITY OF NOTRE DAME

1 *Select the correct response to each of the following questions.*

i. Which of the following would be best described as *"a function that belongs to an object?"*

- ○ *a lambda*
- ⊗ *a method*
- ○ *a class*
- ○ *an attribute*

ii. Which of the following methods implements `+` for a class?

- ⊗ `__add__`
- ○ `__str__`
- ○ `__init__`
- ○ `__iter__`

iii. Which of the following produces a *syntax error?*

- ○ `[n for n in range(100) if n >= 50]`
- ○ `[n in range(100)]`
- ⊗ `[str(n): n + 1 for n in range(100)]`
- ○ `{str(n): n + 1 for n in range(100)}`

this kind of key:value syntax only works with dictionaries, but this has [ ] list brackets.

there was a typo in this question; please disregard it

iv. Which of the following lists contain exactly 50 elements?

- ⊗ `[n for n in range(100) if n >= 50]`
- ○ `[n in range(50)]` ← this one only has 1 element
- ⊗ `list(range(5)) + list(range(45))`
- ○ *all three of the other options*

v. Given an integer list `nice_list`, which of the following returns a new list with the elements squared?

- ○ `[x**2 for x in nice_list]`
- ○ `list({x: x**2 for x in nice_list}.values())`
- ○ `list(map(lambda x: x*x, nice_list))`
- ⊗ *all three of the other options*

there was a typo in this problem; please disregard it

vi. Which of the following is *not true* about `dict` objects?

- ○ `dict` *objects have a length*
- ○ `dict` *objects can be empty*
- ○ `dict` *objects are mutable*
- ○ `dict` *objects support iteration*

} all of these are true about dicts

vii. Which of the following adjectives describes `list` objects but not `tuple` objects?

- ○ *iterable*
- ⊗ *mutable*
- ○ *ordered*
- ○ *sortable*

lists are mutable, but tuples are not.

viii. Which of the following keywords is used with *"context managers."*

- ○ `lambda`
- ○ `in`
- ○ `for`
- ⊗ `with`

ix. Which of the following types would Santa use for keeping track of which kids are naughty or nice?

- ⊗ `dict` ← a dict mapping childrens' names (or other identifier) to their naughty/nice status.
- ○ `list`
- ○ `str`
- ○ `tuple`

x. Which of the following types would Santa use to keep track of the houses he still needs to visit?

- ○ `dict`
- ⊗ `list`
- ○ `str`
- ○ `tuple`

In order to do the "same" thing with a tuple, Santa would have to create a new tuple with one less element every time he visited a house.

Although I would accept either answer, a list would be preferable because houses that have already been visited should be removed from the collection. Tuples would not allow the removal of elements.

NAME: NETID:

2 *Answer the following questions based on the dataset below.*

```
name,valence
Abuela,Nice
Aiko,Naughty
Albert,Naughty
Alberta,Nice
Brody,Nice
Ezra,Nice
Mary-Alice,Nice
Maximus,Nice
Vita,Naughty
Vito,Nice
```

The dataset is in a file whose relative path is given by `data/naughty-or-nice.csv` for each of the following questions.

i. What does the block of code below display in the terminal?

```
1 with open("data/naughty-or-nice.csv", "r") as file:
2   for line in file:
3     print(1 if line.strip().split(",")[-1] == "Nice" else 0)
```

Notice that the first line of the file is not skipped.

0
1
0
1
1
1
0
1

ii. What is the *value* of `verdict` after running the code below?

```
1 judgement = {"Nice": 0, "Naughty": 0}
2 with open("data/naughty-or-nice.csv", "r") as file:
3   next(file)
4   for line in file:
5     judgement[line.strip().split(",")[1]] += 1
6 verdict = sum(judgement.values())//len(judgement)
```

at the end:

judgement = {"Nice": 7, "Naughty": 3}, len(judgement) = 2

5

iii. What is the *value* of `present` after running the code below?

```
1 info = {}
2 with open("data/naughty-or-nice.csv", "r") as file:
3   for line in list(file)[1:]:
4     elf = line.split(",")[0][0]
5     info[elf] = 1 if elf not in info else info[elf] + 1
6 present = max(info.keys(), key=lambda x: info[x])
```

the first char on each line sets value to 1 or increments by 1 depending on whether that key is already in the dict

returns the key (i.e., the first char of the line) which occurred the most

"A"

iv. What is the *value* of `fireplace` after running the code below?

```
1 with open("data/naughty-or-nice.csv", "r") as file:
2   next(file)
3   log = [line.strip().split(",") for line in file]
4 fire = list(map(lambda x: len(x[0]) + len(x[1]), log))
5 fireplace = max(enumerate(fire), key=lambda x: x[1])[0]
```

log = [["Abuela", "Nice"], ["Aiko", "Naughty"], ...]

6

on line 4, we take the function $\lambda(x) = \text{len}(x[0]) + \text{len}(x[1])$ and apply it to all of the elements of log.

for example, λ(["Abuela", "Nice"]) = len("Abuela") + len("Nice") = 10.

the result is fire = [10, 11, 13, 11, 9, 8, 14, 10, 11, 8].

Then, the max on line 5 returns (6, 14), the first element of which is 6.

So, fireplace = 6.

at the end, the dictionary looks like: info = {"A": 4, "B": 1, "E": 1, "M": 2, "V": 2}

the key with the max value is "A".

NAME: NETID:

3 *Answer the following questions based on the class below.*

```
class Point:
  def __init__(self, x, y):
    self.x = x
    self.y = y
  def __eq__(self, point):
    if isinstance(point, Point):
      return (self.x == point.x) and (self.y == point.y)
    if isinstance(point, tuple) or isinstance(point, list):
      return self.x == point[0] and self.y == point[1]
  def __add__(self, point):
    return Point(self.x + point.x, self.y + point.y)
  def __sub__(self, point):
    return Point(self.x - point.x, self.y - point.y)
  def __mult__(self, other):
    if isinstance(other, int) or isinstance(other, float):
      return Point(scalar*self.x, scalar*self.y)
(!) elif isinstance(other, Point):
      return self.x*other.x + self.y*other.y
  def __str__(self):
    return f"({self.x}, {self.y})"
```

the init method runs whenever an object of this class is created; it initializes the object by setting the two attributes x, y.

this method determines how == works with Point objects (and thus, != also).

this method defines how + works with Points

this method defines how - works with Points

this method defines how * interacts with Points.

this method determines how Point objects display as strings.

i. What does the following block of code display in the terminal?

```
1 snowball = Point(0, 0)
2 trajectory = Point(1, 1)
3 for i in range(5):
4   snowball = snowball + trajectory
5   trajectory = trajectory - (0, 0.1)
6 print(snowball)
```

originally, trajectory.x and trajectory.y are both integers. x stays an integer because x-0 is an integer, but y becomes a float because y-0.1 is converted to a float automatically.

this creates a new object Point(...,...), whose x attribute is (snowball.x + trajectory.x) and whose y attribute is (snowball.y + trajectory.y).

same as above, this creates a new Point with the same x but with y decremented by 0.1.

(5, 4.0)

ii. What does the following block of code display in the terminal?

```
1 rudolf = Point(0, 0)
2 rudolf.x, rudolf.y = (5, 2)
3 print(rudolf*rudolf)
```

← the values 5 and 2 are unpacked, so this executes as:
rudolf.x = 5
rudolf.y = 2

Looking at the (!) lines highlighted above, this computes:

(rudolf.x * rudolf.x) + (rudolf.y * rudolf.y) = (5*5) + (2*2)
= 29

29

iii. The code below throws an error; on *what line* does it occur?

```
1 north_pole = (0, 0)
2 santa = Point(64, 32)
3 cookies = 0
4 milk = 0
5 while not santa == north_pole:
6   cookies += santa.x + santa.y
7   milk += (santa*2).y - santa.x
8   santa = santa//2
```

line 8

santa is a Point type object, but we have not implemented a way for Points to interact with the // operator. to do that, we would need to implement the __floordiv__ method.
Similarly, the __div__ method defines how to interact with /.

NAME: NETID:

4 *Program a solution to the following question in Python.*

You are given a dataset `map.csv` formatted according to the rules below.

1. The first line is a string consisting of the characters `"L"` and `"R"`.
2. The second line is a *token* representing a *starting location*.[1]
3. Every line afterwards consists contains three *tokens*:
   (a) The first token represents a *start* location.
   (b) The second and third tokens represent *left* and *right* destinations.
   (c) The first token is separated from the other tokens by `" = "`.
   (d) The second and third tokens are surrounded by parentheses.
   (e) The second and third tokens are separated by `", "`.
   (f) First tokens will never be repeated.

RESTRICTIONS:
- No use of `import` statements.
- No use of functions, methods, nor types that we have not covered in class nor problem sets.

[1] A *token* is a string of three consecutive upper-case letters.

These lines represent transitions An example dataset is given below.

```
LRRL
AAA
AAA = (BBB, CCC)
BBB = (DDD, EEE)
CCC = (ZZZ, GGG)
DDD = (DDD, EEE)
EEE = (BBB, ZZZ)
GGG = (GGG, GGG)
ZZZ = (ZZZ, ZZZ)
```

Your task is traverse the map beginning at the starting location[2] by following the `"L"` and `"R"` directions[3] and seeing where your traversal ends. In the directions, an `"L"` signifies that you should go to the *second* token,[4] and an `"R"` signifies that you should go to the *third* token.[5] In the example dataset above, our final destination would be `ZZZ`.

[2] given by the token on the second line
[3] given by the string on the first line
[4] the token on the *left* of the ordered pair
[5] the token on the *right* of the ordered pair

$$\text{AAA} \xrightarrow{L} \text{BBB} \xrightarrow{R} \text{EEE} \xrightarrow{R} \text{ZZZ} \xrightarrow{L} \text{ZZZ}$$

Write a block of code that reads this file and prints the final destination of reached by traversing the map according to the directions.

NOTE: you may name your function and your variables whatever you'd like.

```
with open("map.csv", "r") as file:
    directions = next(file).strip()
    start = next(file).strip()
    chart = { line[0:3] : (line[7:10], line[12:15]) for line in file }

location = start
for char in directions:
    if char == "L":
        location = chart[location][0]
    else:
        location = chart[location][1]

print(location)
```

4 / 4